Elasticsearch：Elasticsearch中的refresh和flush操作指南

在今天的文章里，我们来主要介绍一下Elasticsearch的refresh及flush两种操作的区别。如果我们从字面的意思上讲，好像都是刷新的意思。但是在Elasticsearch中，这两种操作是有非常大的区别的。本指南将有效解决两者之间的差异。 我们还将介绍Lucene功能的基础知识，例如重新打开（reopen)和提交(commit)，这有助于理解refresh和flush操作。

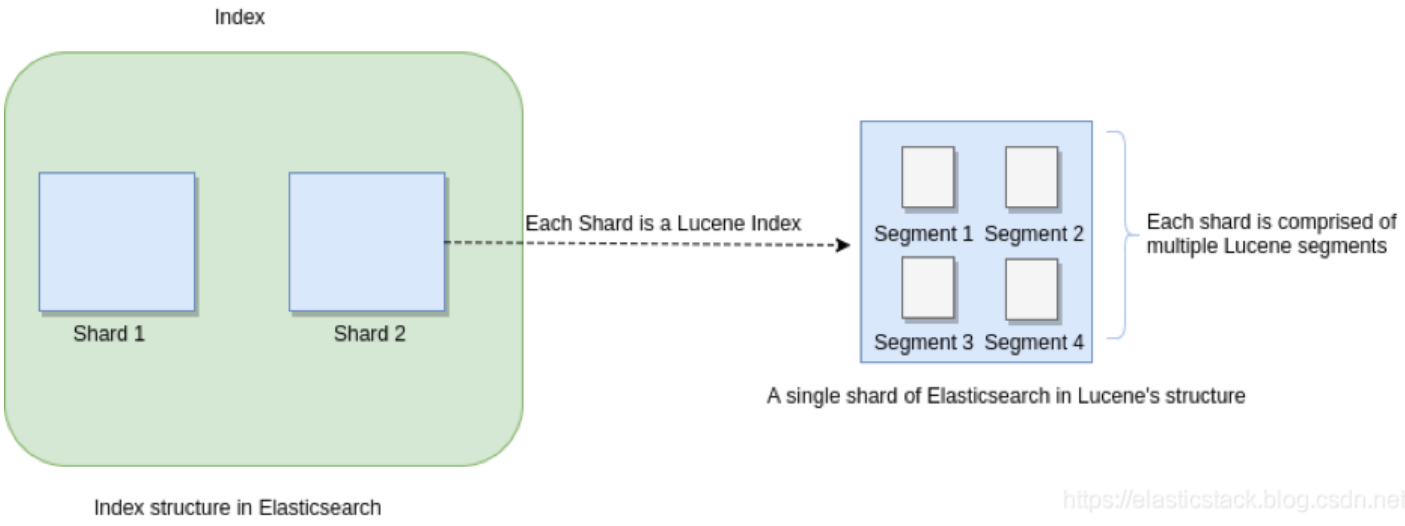
Refresh及Flush

乍一看，Refresh和Flush操作的通用目的似乎是相同的。 两者都用于使文档在索引操作后立即可供搜索。 在Elasticsearch中添加新文档时，我们可以对索引调用_refresh或_flush操作，以使新文档可用于搜索。 要了解这些操作的工作方式，您必须熟悉Lucene中的Segments，Reopen和Commits。Apache Lucene是Elasticsearch中的基础查询引擎。

Lucene中的Segments

在Elasticsearch中，最基本的数据存储单位是shard。 但是，通过Lucene镜头看，情况会有所不同。 在这里，每个Elasticsearch分片都是一个Lucene索引(index)，每个Lucene索引都包含几个Lucene segments。 一个Segment包含映射到文档里的所有术语（terms)一个反向索引(inverted index)。

下图显示了段的概念及其如何应用于Elasticsearch索引及其分片：



这种分Segment的概念是，每当创建新文档时，它们就会被写入新的Segment中。 每当创建新文档时，它们都属于一个新的Segment，并且无需修改前一个Segment。 如果必须删除文档，则在其原始Segment中将其标记为已删除。 这意味着它永远不会从Segement中物理删除。

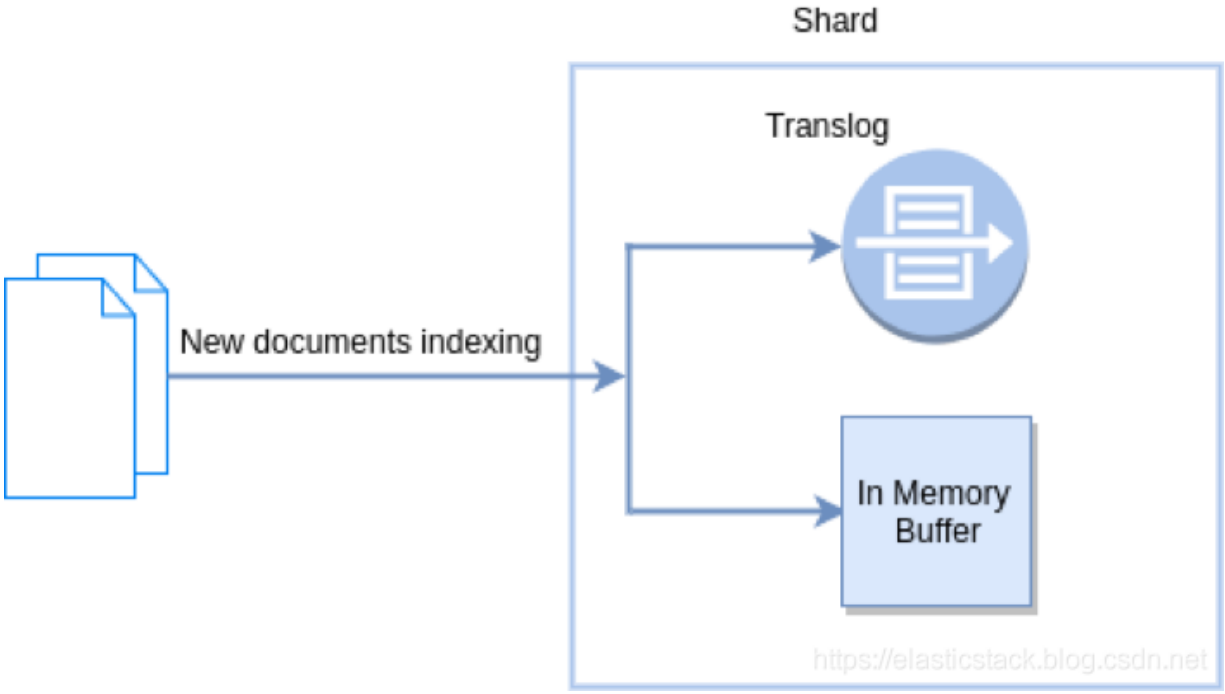
与更新相同：文档的先前版本在上一个Segment中被标记为已删除，更新后的版本保留在当前Segment中的同一文档ID下。

Lucene中的Reopen

当调用Lucene Reopen时，将使累积的数据可用于搜索。 尽管可以搜索最新数据，但这不能保证数据的持久性或未将其写入磁盘。我们可以调用n次重新打开功能，并使最新数据可搜索，但不能确定磁盘上是否存在数据。

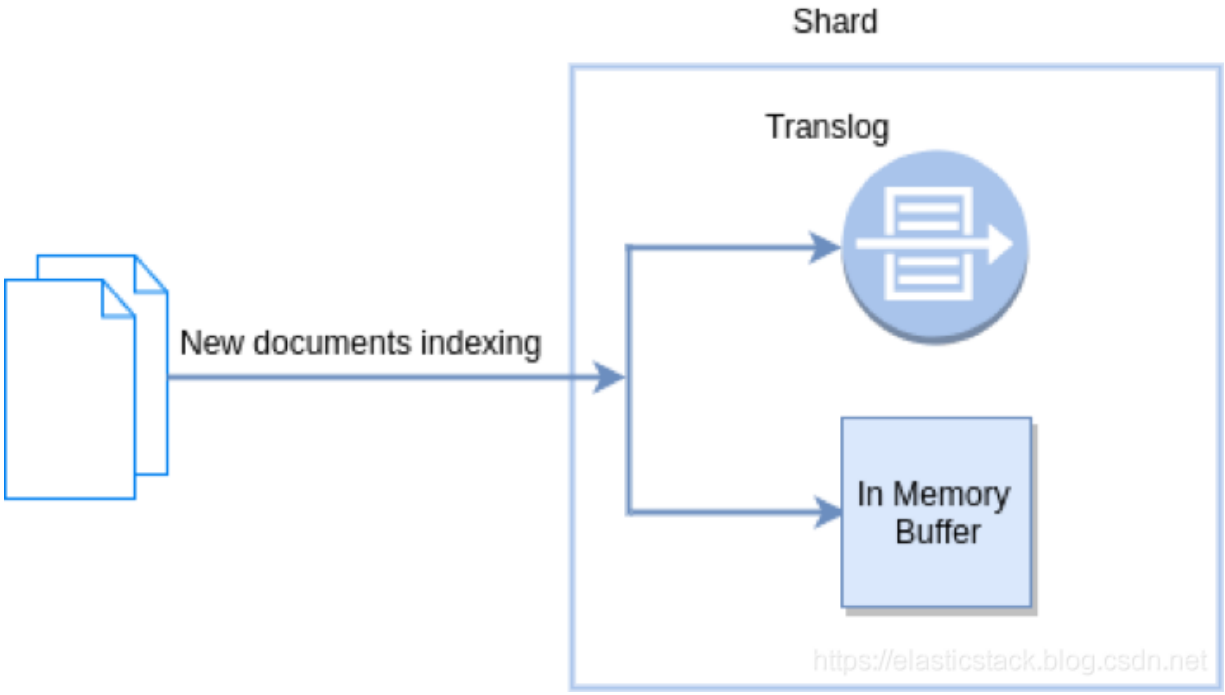
Lucene中的Commits

Lucene提交使数据安全。 对于每次提交，来自不同段的数据将合并并推送到磁盘，从而使数据持久化。 尽管提交是持久保存数据的理想方法，但问题是每个提交操作都占用大量资源。 每个提交操作都有其自己的内部 I/O 操作以及与其相关的读/写周期。 这就是为什么我们希望在基于Lucene的系统中一次又一次地重新使用重新打开功能以使新数据可搜索的确切原因。



Elasticsearch中的Translog

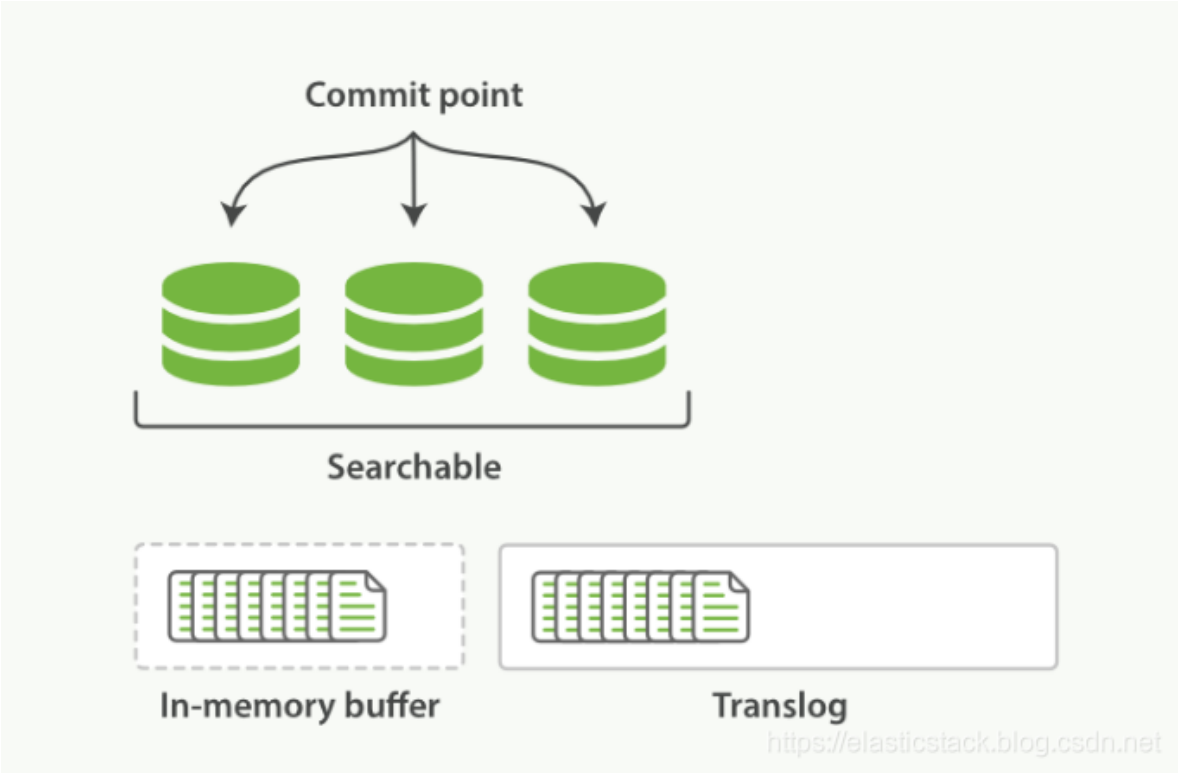
Elasticsearch采用另一种方法来解决持久性问题。 它在每个分片中引入一个事务日志（transaction log）。 已建立索引的新文档将传递到此事务日志和内存缓冲区中。 下图显示了此过程：



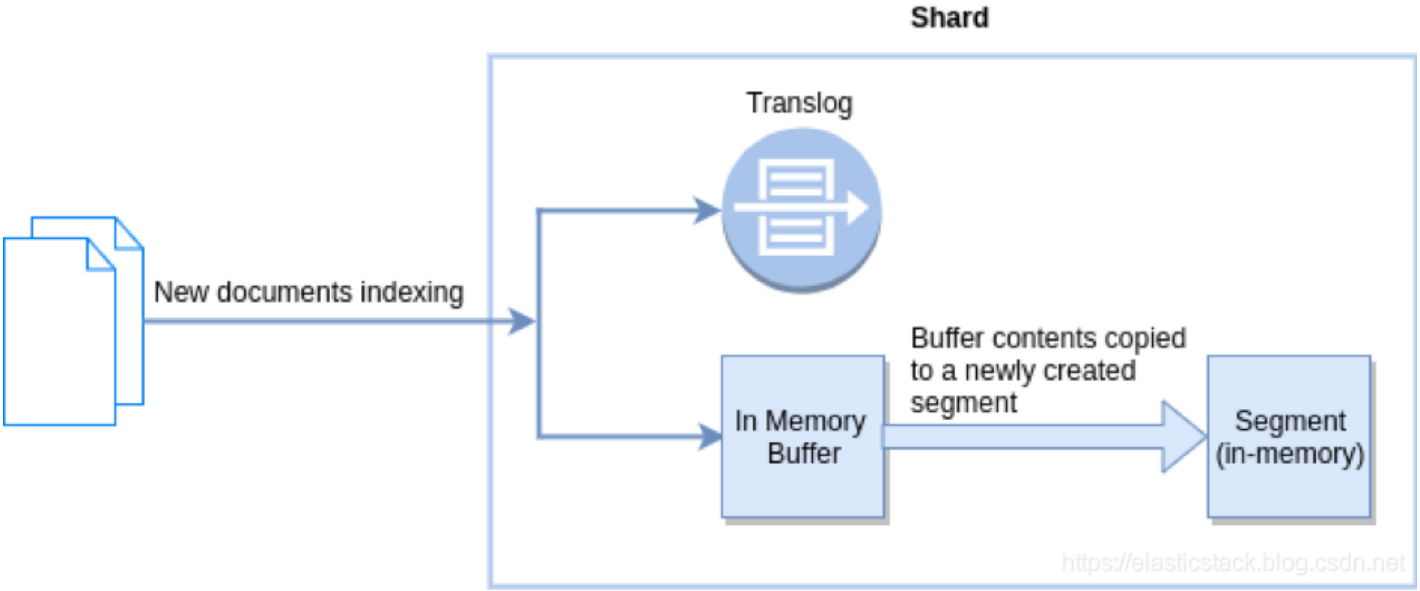
Elasticsearch中的refresh

当我们把一条数据写入到Elasticsearch中后，它并不能马上被用于搜索。新增的索引必须写入到Segment后才能被搜索到，因此我们把数据写入到内存缓冲区之后并不能被搜索到。新增了一条记录时，Elasticsearch会把数据写到translog和in-memory buffer(内

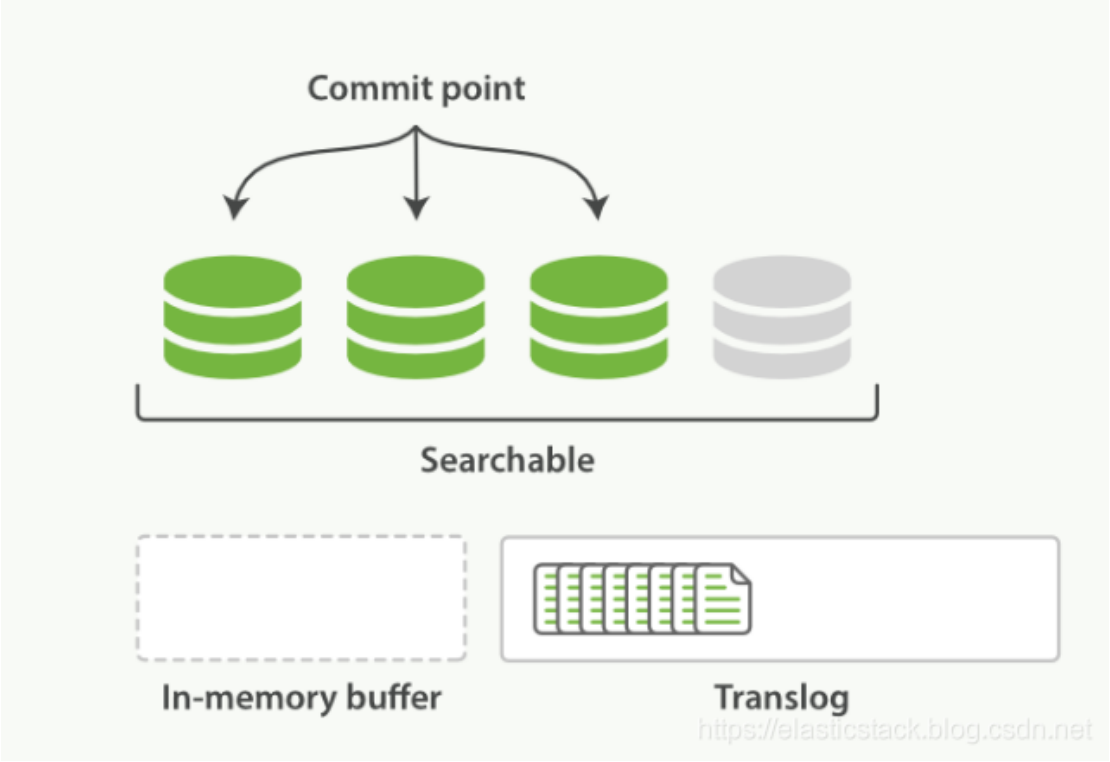
存缓存区)中,如下图所示:



如果希望该文档能立刻被搜索，需要手动调用refresh操作。在Elasticsearch中，默认情况下_refresh操作设置为每秒执行一次。 在此操作期间，内存中缓冲区的内容将复制到内存中新创建的Segment中，如下图所示。 结果，新数据可用于搜索。



这个refresh的时间间隔可以由index设置中index.refresh_interval来定义。执行完refresh后的结果如下：



我们可以看出来，在In-meomory buffer中，现在所有的东西都是空的，但是Translog里还是有东西的。

refresh的开销比较大,我在自己环境上测试10W条记录的场景下refresh一次大概要14ms,因此在批量构建索引时可以把refresh间隔设置成-1来临时关闭refresh,等到索引都提交完成之后再打开refresh,可以通过如下接口修改这个参数:

```
curl -XPUT 'localhost:9200/test/_settings' -d '{
    "index" : {
        "refresh_interval" : "-1"
    }
}'
```

另外当你在做批量索引时,可以考虑把副本数设置成0，因为document从主分片(primary shard)复制到从分片(replica shard)时,从分片也要执行相同的分析、索引和合并过程,这样的开销比较大，你可以在构建索引之后再开启副本，这样只需要把数据从主分片拷贝到从分片:

```
curl -XPUT 'localhost:9200/my_index/_settings' -d ' {
    "index" : {
        "number_of_replicas" : 0
    }
}'
```

执行完批量索引之后,把刷新间隔改回来:

```
curl -XPUT 'localhost:9200/my_index/_settings' -d '{
    "index" : {
        "refresh_interval" : "1s"
    }
}'
```

你还可以强制执行一次refresh以及索引分段的合并:

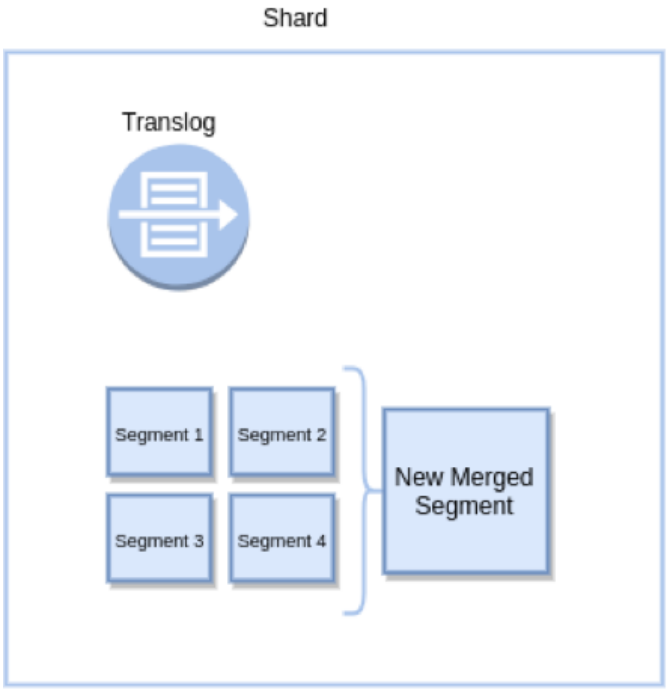
```
curl -XPOST 'localhost:9200/my_index/_refresh'
curl -XPOST 'localhost:9200/my_index/_forcemerge?max_num_segments=5'
```

Translog及持久化存储

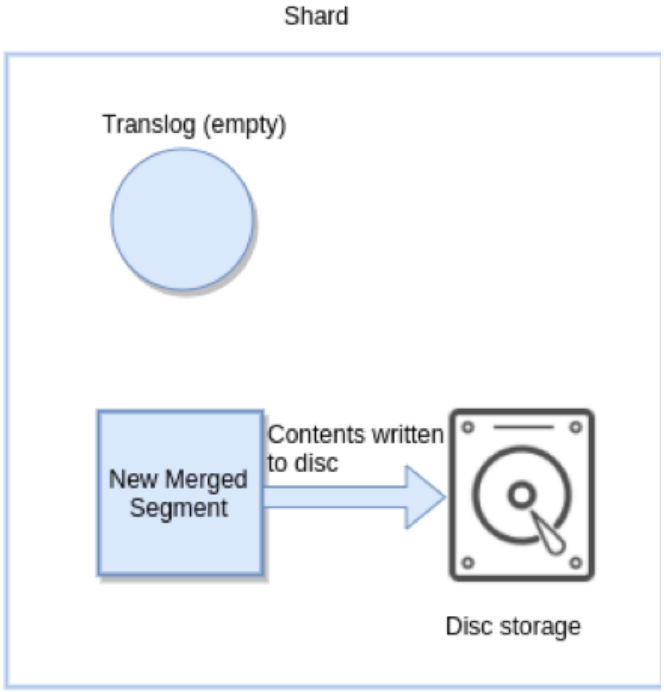
但是，translog如何解决持久性问题？ 每个Shard中都存在一个translog，这意味着它与物理磁盘内存有关。 它是同步且安全的，因此即使对于尚未提交的文档，您也可以获得持久性和持久性。 如果发生问题，可以还原事务日志。 同样，在每个设置的时间间隔内，或在成功完成请求（索引，批量，删除或更新）后，将事务日志提交到磁盘。

Elasticsearch中的Flush

Flush实质上意味着将内存缓冲区中的所有文档都写入新的Lucene Segment，如下面的图所示。 这些连同所有现有的内存段一起被提交到磁盘，该磁盘清除事务日志（参见图4）。 此提交本质上是Lucene提交（commit）。



Stage 1
Figure 3



Stage 2
Figure 4

Flush会定期触发，也可以在Translog达到特定大小时触发。 这些设置可以防止Lucene提交带来的不必要的费用。

结论

在本指南中，我们探索了两个紧密相关的Elasticsearch操作，_flush和_refresh显示了它们之间的共性和差异。 我们还介绍了Lucene的基础架构组件-重新打开（reopen)并提交(commits)-这有助于掌握Elasticsearch中_refresh和_flush操作的要点。

简而言之，_refresh用于使新文档可见以进行搜索。 而_flush用于将内存中的段保留在硬盘上。 `_flush` 不会影响Elasticsearch中文档的可见性，因为搜索是在内存段中进行的，而~~不是~~ `_refresh` 会影响其可见性。